|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Department of Information Technology**  **314454C: Laboratory Practice-II (Cloud Computing)**  IT (2019 Course) Semester- VI | | | | | |
|  | **Teaching Scheme** |  | **Examination Scheme** | |  |
| Practical : | 4 Hrs. / Week | Term work : | 50 Marks |
|  |  | Practical | 25 Marks |
|  | LAB MANUAL  **ACADEMIC YEAR 2022-23** | | | | |
|  | | | | | |

**Vision of the Institute**

We are committed to produce not only good engineers but good Human beings, also.

**Mission of the Institute**

Holistic development of students and teachers is what we believe in and work for. We strive to achieve this by imbibing a

unique value system, transparent work culture, excellent

academic and physical environment conducive to learning,

creativity & technology transfer. Our mandate is to generate,

preserve and share knowledge for developing a vibrant Society.

**Vision of the Department**

To educate students with quality-based IT education that quips

them to become creative, successful professionals for the betterment of the Society**.**

**Mission of the Department**

To offer graduate program in IT for making students,

employable, professional and encouraging them for higher

studies and research. To create successful graduates who are committed to lifelong learning with personal, professional & social responsibility

### 

## Third Year Information Technology (2019 Course)

## 314454C: Laboratory Practice-II (Cloud Computing)

#### Prerequisite Courses:

* Basics of Computer Networks
* Operating Systems

#### Course Objectives:

1. To provide students with the fundamentals and essentials of cloud computing

2. To learn basics of virtualization and its importance

3. To provide students a sound foundation of the cloud computing so that they are able to start using

and adopting cloud computing services and tools in their real-life scenarios

4. To enable students exploring some important cloud computing driven commercial systems and

applications

5. To understand cloud storage technologies and relevant file systems

6. To be exposed to Ubiquitous Cloud and Internet of Things

**Course Outcomes**:

On completion of the course, students will be able to–

**CO1:** Articulate the main concepts, key technologies and fundamentals of cloud computing.

**CO2:** Understand cloud enabling technologies and virtualization.

**CO3:** Analyze various cloud programming models and apply them to solve problems on the cloud.

**CO4:** Explain data storage and major security issues in the cloud.

**CO5:** Understand trends in ubiquitous cloud and internet of things.

**CO6:** Explore future trends of cloud computing.

#### LIST OF ASSIGNMENTS

1. Install Google App Engine. Create hello world app and other simple web applications using python/java.
2. Use GAE launcher to launch the web applications.
3. Simulate a cloud scenario using Cloud Sim and run a scheduling algorithm that is not present in Cloud Sim.
4. Find a procedure to transfer the files from one virtual machine to another virtual machine.
5. Find a procedure to launch virtual machine using try stack (Online Open stack Demo Version)
6. Design and deploy a web application in a PaaS environment.
7. Design and develop custom Application (Mini Project) using Salesforce Cloud.
8. Design an Assignment to retrieve, verify, and store user credentials using Firebase Authentication, the Google App Engine standard environment, and Google Cloud Data store.

#### CASE STUDIES

* Data storage security in private cloud
* Application of IoT/Ubiquitous based on cloud
* Tools for building private cloud

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr.**  **No.** | **Title** | **Page No.** |
| 1 | Install Google App Engine. Create hello world app and other simple web applications using python/java. | 14 |
| 2 | Use GAE launcher to launch the web applications. | 31 |
| 3 | Simulate a cloud scenario using Cloud Sim and run a scheduling algorithm that is not present in Cloud Sim. | 52 |
| 4 | Find a procedure to transfer the files from one virtual machine to another virtual machine. | 69 |
| 5 | Find a procedure to launch virtual machine using try stack (Online Open stack Demo Version) | 70 |
| 6 | Design and deploy a web application in a PaaS environment. | 74 |
| 7 | Design and develop custom Application (Mini Project) using Salesforce Cloud. | 78 |
| 8 | Design an Assignment to retrieve, verify, and store user credentials using Firebase Authentication, the Google App Engine standard environment, and Google Cloud Data store. | - |
| 9 | Case Study: Data storage security in private cloud | - |
| 10 | Case Study: Application of IoT/Ubiquitous based on cloud | - |
| 11 | Case Study: Tools for building private cloud | 91 |

**Lab Planning (Scheduling)**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No | Title | No. of Hrs | Week |
| 1 | Install Google App Engine. Create hello world app and other simple  web applications using python/java. |  |  |
| 2 | Use GAE launcher to launch the web applications. |  |  |
| 3 | Simulate a cloud scenario using Cloud Sim and run a scheduling algorithm that is not present in  Cloud Sim. |  |  |
| 4 | Find a procedure to transfer the files from one virtual machine to  another virtual machine. |  |  |
| 5 | Find a procedure to launch virtual machine using try stack (Online Open stack Demo  Version) |  |  |
| 6 | Design and deploy a web application in a PaaS environment. |  |  |
| 7 | Design and develop custom Application (Mini Project) using Salesforce Cloud. |  |  |
| 8 | Design an Assignment to retrieve, verify, and store user credentials using Firebase Authentication, the Google App Engine standard environment, and Google Cloud  Data store. |  |  |
| 9 | Case Study: Data storage security in private cloud |  |  |
| 10 | Case Study: Application of IoT/Ubiquitous based on cloud |  |  |
| 11 | Case Study: Tools for building private cloud |  |  |

**Assignment No. 1**

**Date:**

**Title:** Install Google App Engine. Create hello world app and other simple web applications

using python/java.

###### Assignment No. 1

**Aim:** Install Google App Engine. Create hello world app and other simple web applications using python/java.

###### Objective:

* + Installing Google App Engine.

###### Theory:

**Install an SDK for App Engine**

To set up your environment for developing on Python 3:

* 1. Install Python 3 by downloading it from the official site.
  2. Install the python setup into your system.
  3. Open CMD and type in python or python --version to check if it has been installed properly.
  4. If the CMD is not recognizing the commands then check the system variables and set the path of python to the respective directory.

Creating your Google cloud account.

1. Create a google account.
2. Go to the Google Cloud Website ‘https://cloud.google.com/appengine/’ and create the G Cloud account there.
3. On successful creation of the account download the python or Java SDK for G Cloud CLI from

For Python: <https://dl.google.com/dl/cloudsdk/channels/rapid/GoogleCloudSDKInstaller.exe>

1. Download the respective SDK and install it on the local machine. Creating the first project on google cloud.
2. First, create a folder where you want to make the app.
3. Then first create an app.yaml file and type the following code in it.

application: your-app-identifier version: 1

runtime: python3.10 api\_version: 1 threadsafe: true

handlers:

-url: /.\*

script: hellowork.app

1. Then create a helloworld.py file and type the following in it.

import webapp2

class MainPage(webapp2.RequestHandler): def get(self):

self.response.headers['Content-Type'] = 'text/plain' self.response.write('Hello World!')

app = webapp2.WSGIApplication([ ('/', MainPage),

], debug=True)

1. Once this is done you are ready with the app.
2. To test the code is working open CMD and type the following

python ‘C:\Program Files (x86)\Google\google\_appengine\dev\_appserver.py’

\path\to\helloworld

This will start the app in a local environment.

1. Then go to http://localhost:8080 to see the app print ‘Hello World!’.
2. To deploy the app on the google cloud server type the following commands.

python ‘C:\Program Files (x86)\Google\google\_appengine\appcfg.py’ update

\path\to\helloworld

This might ask for the Google Cloud credentials for uploading the app.

Note: The app when hosted on Google Cloud might charge for services on normal ID so do turn off the app after creating it and remove it from the cloud.

###### Output:

Successfully created the first app and hosted it on GCloud.

###### Conclusion:

We learned how to create the app and deploy it to Google Cloud.

###### FAQ:

**Assignment No. 2 Aim:** Use GAE launcher to launch the web applications.

###### Objective:

Creating and deploying an application on GAE

###### Theory:

Creating the first project on google cloud.

1. First, create a folder where you want to make the app.
2. Then first create an app.yaml file and type the following code in it.
3. Then create a the app that you want to upload on the Gcloud platform.
4. Once this is done you are ready with the app.
5. To test the code is working open CMD and type the following

python ‘C:\Program Files (x86)\Google\google\_appengine\dev\_appserver.py’

\path\to\helloworld

This will start the app in a local environment.

1. Then go to http://localhost:8080 to see the app print ‘Hello World!’.
2. To deploy the app on the google cloud server type the following commands.

python ‘C:\Program Files (x86)\Google\google\_appengine\appcfg.py’ update

\path\to\helloworld

This might ask for the Google Cloud credentials for uploading the app.

Note: The app when hosted on Google Cloud might charge for services on normal ID so do turn off the app after creating it and remove it from the cloud.

###### Output:

You successfully developed the app and hosted it on Gcloud platform.

###### Conclusion:

Learned how to host app on Gcloud platform.

###### FAQ:

**Assignment No. 3**

**Aim:** Simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim.

###### Objective:

* + Install CloudSim on system.
  + Run Scheduling algorithm that is not present in CloudSim.

###### Theory:

What is CloudSim?

CloudSim is an open-source framework, which is used to simulate cloud computing infrastructure and services. It is developed by the CLOUDS Lab organization and is written entirely in Java. It is used for modelling and simulating a cloud computing environment as a means for evaluating a hypothesis prior to software development in order to reproduce tests and results.

##### Benefits of Simulation over the Actual Deployment:

Following are the benefits of CloudSim:

##### No capital investment involved. With a simulation tool like CloudSim there is no installation or maintenance cost.

* + - Easy to use and Scalable. You can change the requirements such as adding or deleting resources by changing just a few lines of code.

##### Risks can be evaluated at an earlier stage. In Cloud Computing utilization of real testbeds limits the experiments to the scale of the testbed and makes the reproduction of results an extremely difficult undertaking. With simulation, you can test your product against test cases and resolve issues before actual deployment without any limitations.

* + - No need for try-and-error approaches. Instead of relying on theoretical and imprecise evaluations which can lead to inefficient service performance and revenue generation, you can test your services in a repeatable and controlled environment free of cost with CloudSim.

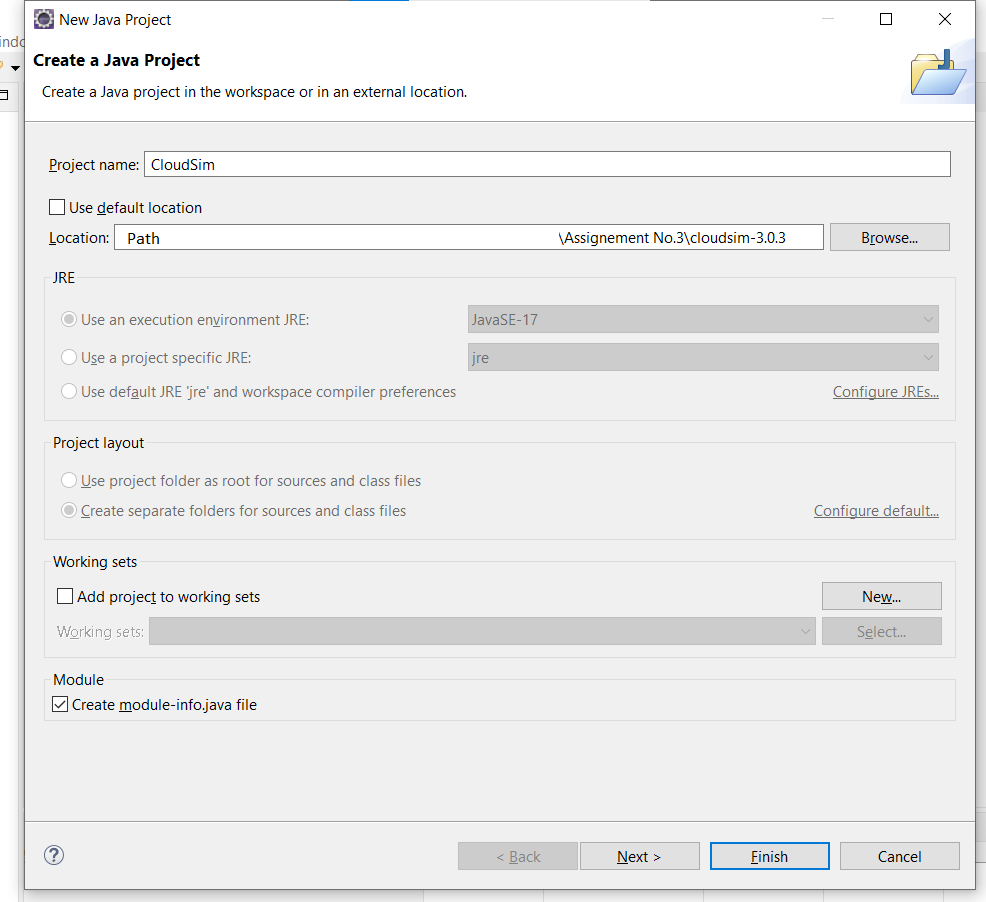
Pre-requistes:

* + Knowledge of OOP and Java Collections.
  + Basics of cloud computing. Installation

1. Download CloudSim from the provided link. <https://github.com/Cloudslab/cloudsim/releases>

Download the common-math3 library binary zip file: <http://commons.apache.org/proper/commons-math/download_math.cgi>

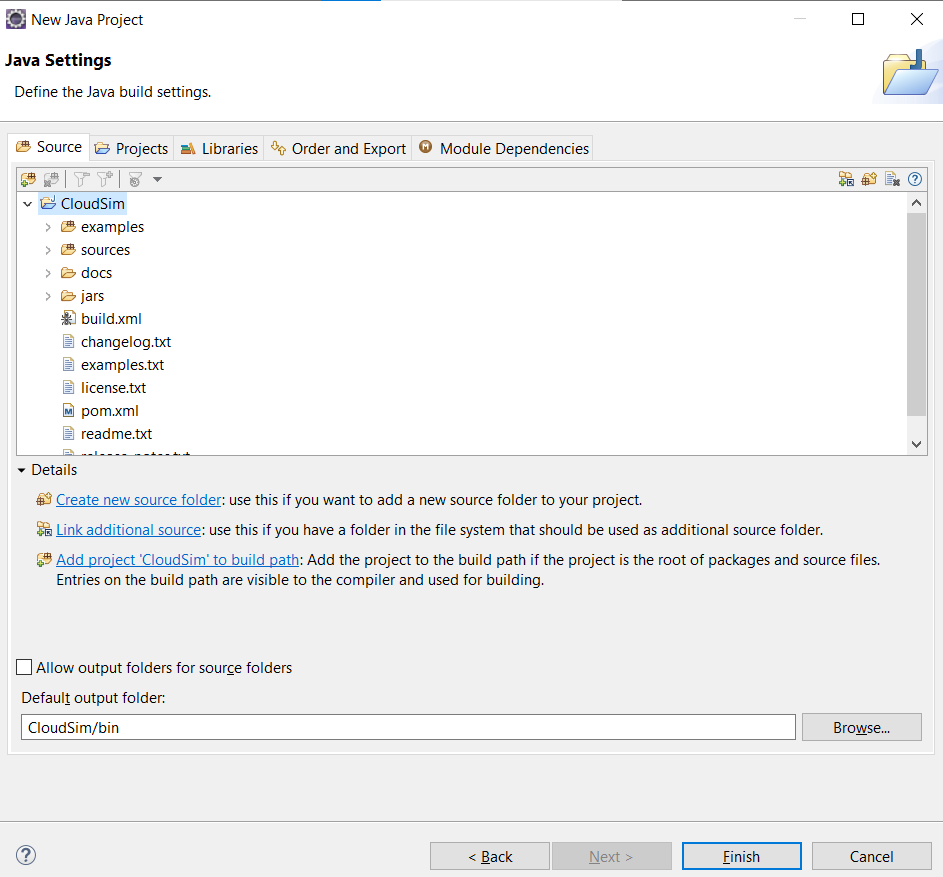
1. Extract the zip file. And also extract commons-math3.3-6 into the same folder.
2. Open Eclipse IDE and go to File->New->Java Project



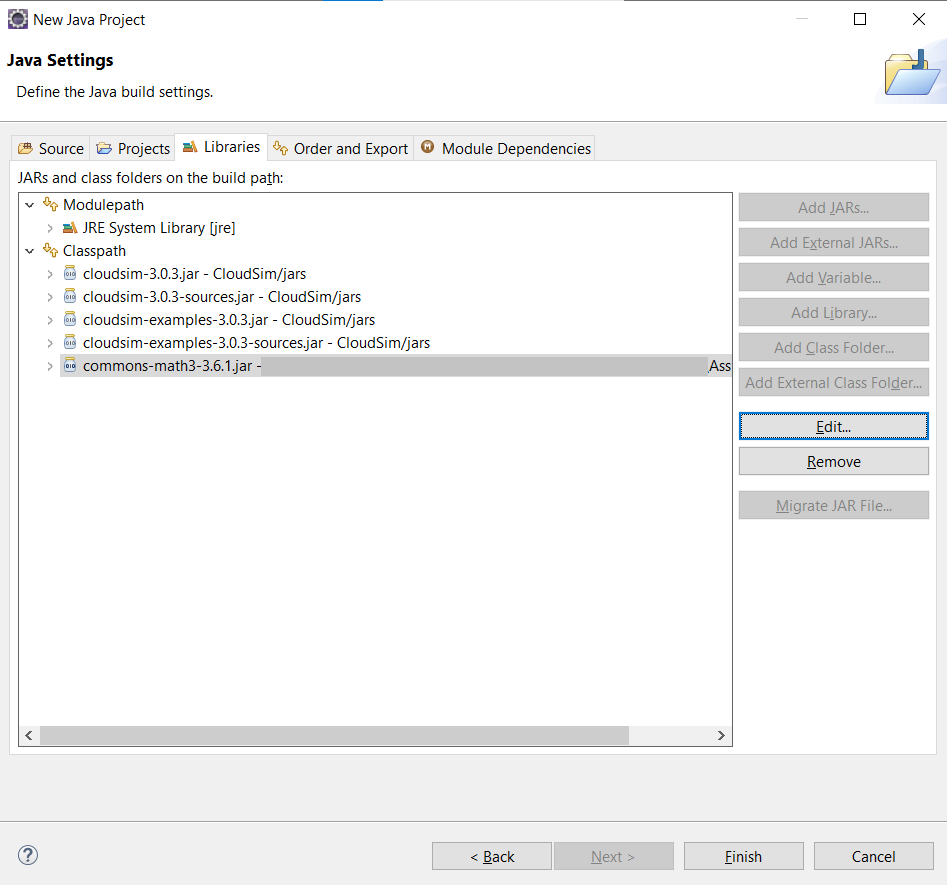
1. Enter the name of the project and then uncheck the use default locations box just under it and click on Browser.

Browse to the folder where you extracted.

1. Click on “Next”.
2. Open the libraries.



1. If you don’t find commons.math3.3.x.jar, click on add external jar, and add the common- math3.3.x.jar file.



1. After that click on finish. It will take 2 to 3 minutes to configure. Now you can explore the project file.

Once the project as been opended in the eclipse, create a new java file where you can write the code for scheduling algorithm.

Once the algorithm has been developed. Create a file to implement the algorithm. Run the implementation file, in the eclipse window.

Source Code: [https://github.com/suyash-more/Cloud-Computing-Projects/tree/master/Scheduling-](https://github.com/suyash-more/Cloud-Computing-Projects/tree/master/Scheduling-Algorithm-in-CloudSim/src) [Algorithm-in-CloudSim/src](https://github.com/suyash-more/Cloud-Computing-Projects/tree/master/Scheduling-Algorithm-in-CloudSim/src)

Execute file SJF\_Scheduler.java

**Output:** The Scheduling algorithm was implemented in the CloudSim environment for managing the tasks.

Starting SJF Scheduler... Initializing new Matrices... Initialising...

Starting CloudSim version 3.0 Datacenter\_0 is starting...

Datacenter\_1 is starting... Datacenter\_2 is starting... Datacenter\_3 is starting... Datacenter\_4 is starting... Broker\_0 is starting...

Entities started.

0.0: Broker\_0: Cloud Resource List received with 5 resource(s) 0.0: Broker\_0: Trying to Create VM #2 in Datacenter\_0

* 1. : Broker\_0: Trying to Create VM #3 in Datacenter\_1 0.0: Broker\_0: Trying to Create VM #4 in Datacenter\_2 0.0: Broker\_0: Trying to Create VM #5 in Datacenter\_3 0.0: Broker\_0: Trying to Create VM #6 in Datacenter\_4
  2. : Broker\_0: VM #2 has been created in Datacenter #2, Host #0

0.1: Broker\_0: VM #3 has been created in Datacenter #3, Host #0

0.1: Broker\_0: VM #4 has been created in Datacenter #4, Host #0

0.1: Broker\_0: VM #5 has been created in Datacenter #5, Host #0

* 1. : Broker\_0: VM #6 has been created in Datacenter #6, Host #0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0.1: | Broker\_0: | Sending | cloudlet | 0 | to | VM | #5 |
| 0.1: | Broker\_0: | Sending | cloudlet | 1 | to | VM | #4 |
| 0.1: | Broker\_0: | Sending | cloudlet | 2 | to | VM | #3 |
| 0.1: | Broker\_0: | Sending | cloudlet | 3 | to | VM | #5 |
| 0.1: | Broker\_0: | Sending | cloudlet | 4 | to | VM | #4 |
| 0.1: | Broker\_0: | Sending | cloudlet | 5 | to | VM | #4 |
| 0.1: | Broker\_0: | Sending | cloudlet | 6 | to | VM | #3 |
| 0.1: | Broker\_0: | Sending | cloudlet | 7 | to | VM | #5 |
| 0.1: | Broker\_0: | Sending | cloudlet | 8 | to | VM | #3 |
| 0.1: | Broker\_0: | Sending | cloudlet | 9 | to | VM | #3 |
| 0.1: | Broker\_0: | Sending | cloudlet | 10 | to | VM | #4 |
| 0.1: | Broker\_0: | Sending | cloudlet | 11 | to | VM | #4 |
| 0.1: | Broker\_0: | Sending | cloudlet | 12 | to | VM | #4 |
| 0.1: | Broker\_0: | Sending | cloudlet | 13 | to | VM | #2 |
| 0.1: | Broker\_0: | Sending | cloudlet | 14 | to | VM | #3 |
| 0.1: | Broker\_0: | Sending | cloudlet | 15 | to | VM | #3 |
| 0.1: | Broker\_0: | Sending | cloudlet | 16 | to | VM | #2 |
| 0.1: | Broker\_0: | Sending | cloudlet | 17 | to | VM | #3 |
| 0.1: | Broker\_0: | Sending | cloudlet | 18 | to | VM | #5 |
| 0.1: | Broker\_0: | Sending | cloudlet | 19 | to | VM | #3 |
| 0.1: | Broker\_0: | Sending | cloudlet | 20 | to | VM | #2 |
| 0.1: | Broker\_0: | Sending | cloudlet | 21 | to | VM | #3 |
| 0.1: | Broker\_0: | Sending | cloudlet | 22 | to | VM | #3 |
| 0.1: | Broker\_0: | Sending | cloudlet | 23 | to | VM | #3 |
| 0.1: | Broker\_0: | Sending | cloudlet | 24 | to | VM | #6 |
| 0.1: | Broker\_0: | Sending | cloudlet | 25 | to | VM | #2 |
| 0.1: | Broker\_0: | Sending | cloudlet | 26 | to | VM | #2 |
| 0.1: | Broker\_0: | Sending | cloudlet | 27 | to | VM | #4 |
| 0.1: | Broker\_0: | Sending | cloudlet | 28 | to | VM | #6 |
| 0.1: | Broker\_0: | Sending | cloudlet | 29 | to | VM | #5 |

1110.964: Broker\_0: Cloudlet 13 received

2185.4159999999997: Broker\_0: Cloudlet 24 received

2230.7799999999997: Broker\_0: Cloudlet 0 received

2630.228: Broker\_0: Cloudlet 16 received

2631.0319999999997: Broker\_0: Cloudlet 1 received

3805.5319999999997: Broker\_0: Cloudlet 2 received

4558.224: Broker\_0: Cloudlet 4 received

5290.448: Broker\_0: Cloudlet 28 received

5443.424: Broker\_0: Cloudlet 20 received

5664.28: Broker\_0: Cloudlet 3 received

5818.4839999999995: Broker\_0: Cloudlet 6 received

5942.736: Broker\_0: Cloudlet 5 received

6761.9: Broker\_0: Cloudlet 10 received

7463.232: Broker\_0: Cloudlet 7 received

8127.948: Broker\_0: Cloudlet 18 received

8377.448: Broker\_0: Cloudlet 25 received

8612.552: Broker\_0: Cloudlet 11 received

9053.771999999999: Broker\_0: Cloudlet 8 received

10803.18: Broker\_0: Cloudlet 29 received

10924.9: Broker\_0: Cloudlet 12 received

11483.92: Broker\_0: Cloudlet 26 received

12474.599999999999: Broker\_0: Cloudlet 9 received

12856.235999999999: Broker\_0: Cloudlet 27 received

15993.463999999998: Broker\_0: Cloudlet 14 received

18118.568: Broker\_0: Cloudlet 15 received

20474.343999999997: Broker\_0: Cloudlet 17 received

23210.483999999997: Broker\_0: Cloudlet 19 received

25641.607999999997: Broker\_0: Cloudlet 21 received

26796.167999999998: Broker\_0: Cloudlet 22 received

29732.215999999997: Broker\_0: Cloudlet 23 received 29732.215999999997: Broker\_0: All Cloudlets executed. Finishing... 29732.215999999997: Broker\_0: Destroying VM #2

29732.215999999997: Broker\_0: Destroying VM #3

29732.215999999997: Broker\_0: Destroying VM #4

29732.215999999997: Broker\_0: Destroying VM #5

29732.215999999997: Broker\_0: Destroying VM #6 Broker\_0 is shutting down...

Simulation: No more future events

CloudInformationService: Notify all CloudSim entities for shutting down. Datacenter\_0 is shutting down...

Datacenter\_1 is shutting down... Datacenter\_2 is shutting down... Datacenter\_3 is shutting down... Datacenter\_4 is shutting down... Broker\_0 is shutting down...

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Simulation  Simulation  ========== | completed.  completed.  OUTPUT ========== |  | | | | |
| Cloudlet ID | STATUS Data center | ID | VM | ID | Time | Start Time |
| Finish Time | Waiting Time |  |  |  |  |  |
| 13 | SUCCESS 02 |  | 02 |  | 1110.86 | 00.1 |
| 1110.96 | 00 |  |  |  |  |  |
| 24 | SUCCESS 06 |  | 06 |  | 2185.32 | 00.1 |
| 2185.42 | 00 |  |  |  |  |  |
| 00 | SUCCESS 05 |  | 05 |  | 2230.68 | 00.1 |
| 2230.78 | 00 |  |  |  |  |  |
| 16 | SUCCESS 02 |  | 02 |  | 1519.26 | 1110.96 |
| 2630.23 | 1110.86 |  |  |  |  |  |
| 01 | SUCCESS 04 |  | 04 |  | 2630.93 | 00.1 |
| 2631.03 | 00 |  |  |  |  |  |
| 02 | SUCCESS 03 |  | 03 |  | 3805.43 | 00.1 |
| 3805.53 | 00 |  |  |  |  |  |

04 SUCCESS

|  |  |  |  |
| --- | --- | --- | --- |
| 04 | 04 | 1927.19 | 2631.03 |
| 06 | 06 | 3105.03 | 2185.42 |
| 02 | 02 | 2813.2 | 2630.23 |
| 05 | 05 | 3433.5 | 2230.78 |
| 03 | 03 | 2012.95 | 3805.53 |
| 04 | 04 | 1384.51 | 4558.22 |
| 04 | 04 | 819.16 5942.74 | |
| 05 | 05 | 1798.95 5664.28 | |
| 05 | 05 | 664.72 7463.23 | |
| 02 | 02 | 2934.02 | 5443.42 |
| 04 | 04 | 1850.65 | 6761.9 |
| 03 | 03 | 3235.29 | 5818.48 |
| 05 | 05 | 2675.23 | 8127.95 |
| 04 | 04 | 2312.35 | 8612.55 |
| 02 | 02 | 3106.47 | 8377.45 |
| 03 | 03 | 3420.83 | 9053.77 |
| 04 | 04 | 1931.34 | 10924.9 |
| 03 | 03 | 3518.86 | 12474.6 |
| 03 | 03 | 2125.1 | 15993.46 |
| 03 | 03 | 2355.78 | 18118.57 |
| 03 | 03 | 2736.14 | 20474.34 |
| 03 | 03 | 2431.12 | 23210.48 |
| 03 | 03 | 1154.56 | 25641.61 |
| 03 | 03 | 2936.05 | 26796.17 |

4558.22 2630.93

28 SUCCESS 5290.45 2185.32

20 SUCCESS 5443.42 2630.13

03 SUCCESS

5664.28 2230.68

06 SUCCESS

5818.48 3805.43

05 SUCCESS

5942.74 4558.12

10 SUCCESS 6761.9 5942.64

1. SUCCESS

7463.23 5664.18

18 SUCCESS 8127.95 7463.13

25 SUCCESS 8377.45 5443.32

11 SUCCESS 8612.55 6761.8

1. SUCCESS

9053.77 5818.38

29 SUCCESS 10803.18 8127.85

12 SUCCESS 10924.9 8612.45

1. SUCCESS 11483.92 8377.35
2. SUCCESS

12474.6 9053.67

1. SUCCESS 12856.24 10924.8
2. SUCCESS 15993.46 12474.5
3. SUCCESS 18118.57 15993.36

17 SUCCESS 20474.34 18118.47

19 SUCCESS 23210.48 20474.24

1. SUCCESS 25641.61 23210.38
2. SUCCESS 26796.17 25641.51
3. SUCCESS 29732.22 26796.07

Makespan using SJF: 6694.725373488874 org.cloudbus.cloudsim.schedulingalgo.SJF\_Scheduler finished!

**Conclusion:** We successfully installed and implemented the Scheduling alogithm that was not in the CloudSim environment.

**Assignmnet No. 4**

**Aim:** Find a procedure to transfer the filesfrom one virtual machine to another virtual machine.

###### Theory:

**SCP**

**scp** (secure copy) command in Linux system is used to copy file(s) between servers in a secure way. The SCP command or secure copy allows secure transferring of files in between the local host and the remote host or between two remote hosts. It uses the same authentication and security as it is used in the Secure Shell (SSH) protocol. SCP is known for its simplicity, security and pre-installed availability.

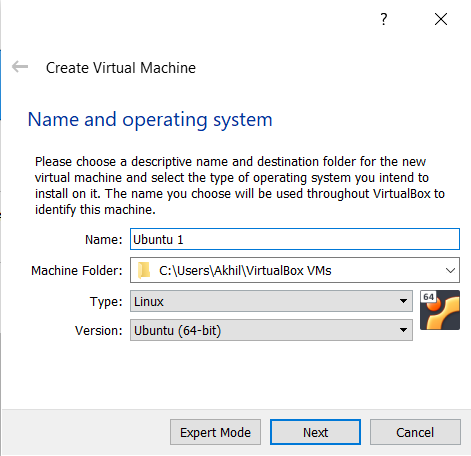
Syntax

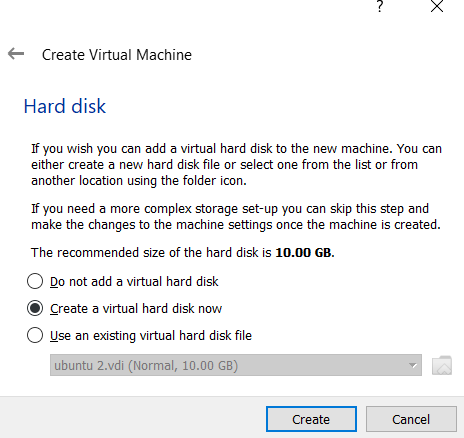
scp [-346BCpqrTv] [-c cipher] [-F ssh\_config] [-i identity\_file] [-l limit] [-o ssh\_option] [-P port] [-S

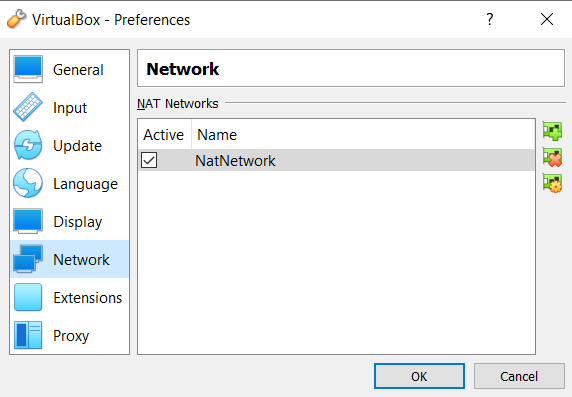
program] [[user@]host1:]file1 … [[user@]host2:]file2

Steps to transfer the files between two files

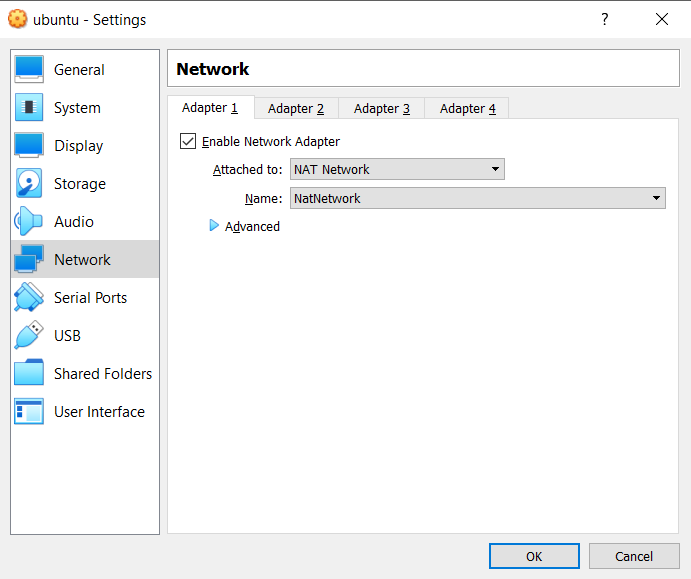
* + 1. Install the virtualbox on our computer.
    2. Download the ubuntu setup and install it or use Virtual Disk Image of the desired ubuntu.
    3. Open the virtual box. Create a new machine.



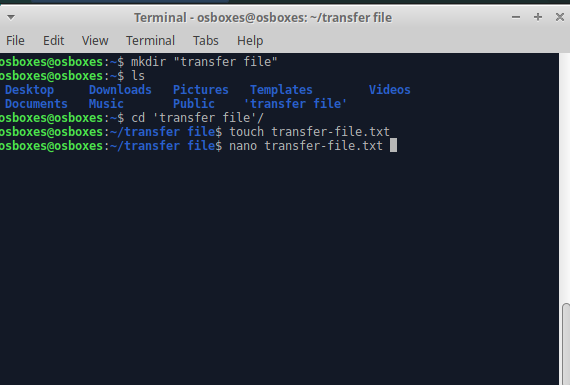
* + 1. Add the basic details and click next.
    2. Then if you are installing the ubuntu, create a new virtual hard disk. Otherwise, use the Virtual disk image as the hard disk for the virtual machine.
    3. Do the same for the second virtual machine too.
    4. Go to file, preferences, and then to network. Then add new network.



* + 1. After that click on the virtual machine, go to settings, then network, and enable the network adapter and attach to NAT network.

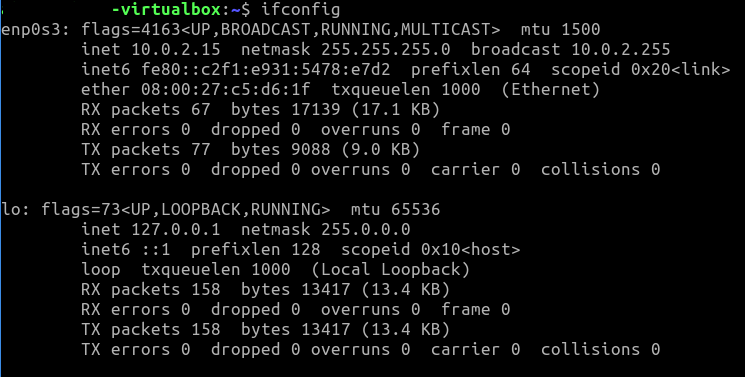


* + 1. Now do the same network setting for the second machine too.
    2. Now start the virtual machines. Open the terminals.
    3. Now chage the directory using mkdir to a desired location where you want to create the new file.
    4. Then to create a file type the following: touch filename.txt



* + 1. Add some data into the file. Then to check if its added properly type the following command. cat filename.txt
    2. Now to transfer the file first of all we need to identify the ip address of the virtual machine to which we want to transfer it. To find it type the ifconfig command. Note the we also need to get the name of the user to whom we want to transfer the files.
    3. Incase if the ifconfig is not installed on your virtual machine install it by the following command.

sudo apt-get install net-tools

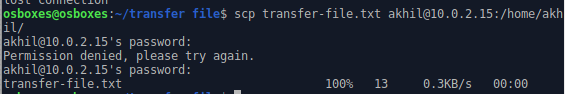


* + 1. Incase if facing any issues type the following commands on terminal of both machines sudo apt-get update

sudo apt-get install openssh-server sudo sfw allow 22

* + 1. Type the following on the terminal of the virtual machine from where you want to transfer the file.

scp filename.txt username@10.0.2.15:/home/username/



* + 1. To check if the file has been successfully transferred, change to the directory, and type ls command.



**Conclusion:** We learnt how to transfer files from one virtual machine to another.

**Assignment No. 5**

**Aim:** Find a procedure to launch a virtual machine using trystack (Online Openstack Demo Version)

###### Theory:

Virtual Machines: VM is no different than any other physical computer like a laptop, smartphone or server. It has a CPU, memory, and disks by which you can store your files and can connect to the internet if needed. In the VM world Operating System running on your computer is called a host and any operating system running inside VMs is called a guest.

Advantages of VMs:

* Cost Saving
* Speed
* Lowered downtime
* Secure Environment
* Access Remotely

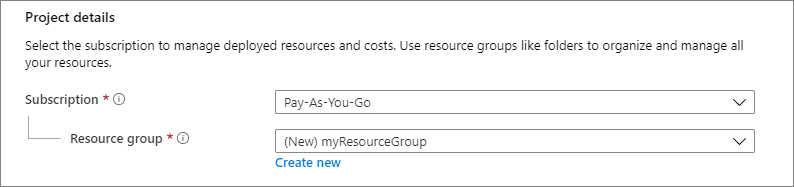
**TryStack:** TryStack is a free and easy way for users to try OpenStack, and setup their cloud with networking, storage and computer instances.

**Requirements:** Account on AWS/ Google Cloud/ Azure

###### Steps:

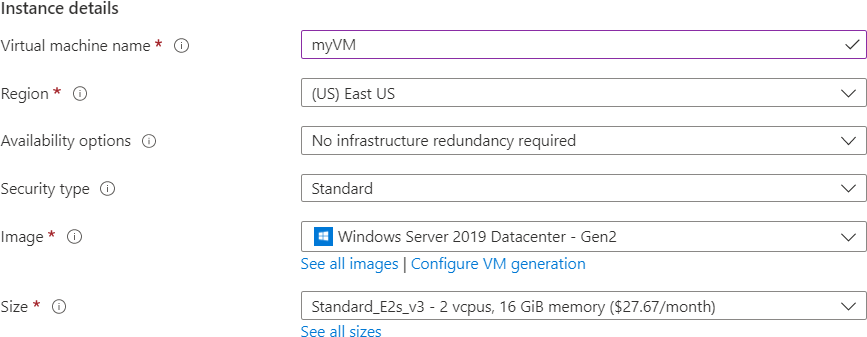
Step1:- Create a virtual machine.

* + - 1. Enter *virtual machines* in the search.
      2. Under Services, select Virtual machines.
      3. In the Virtual machines page, select Create and then Virtual machine. The Create a virtual machine page opens.
      4. In the Basics tab, under Project details, make sure the correct subscription is selected and then choose to Create a new resource group. Enter *myResourceGroup* for the name.



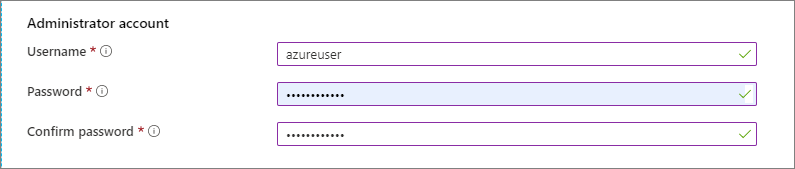
* + - 1. Under **Instance details**, enter *VM name* for the **Virtual machine name** and

choose *Windows Server 2019 Datacenter - Gen2* for the **Image**. Leave the other defaults.

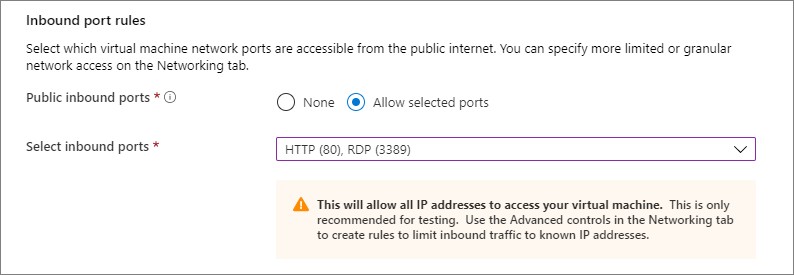


* + - 1. Under the **Administrator account**, select a password, and provide a username, such

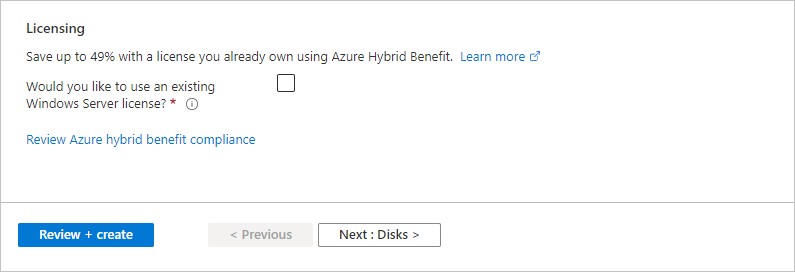
as *azureuser* , and a password. The password must be at least 12 characters long and meet the [defined complexity requirements.](https://docs.microsoft.com/en-us/azure/virtual-machines/windows/faq#what-are-the-password-requirements-when-creating-a-vm-)



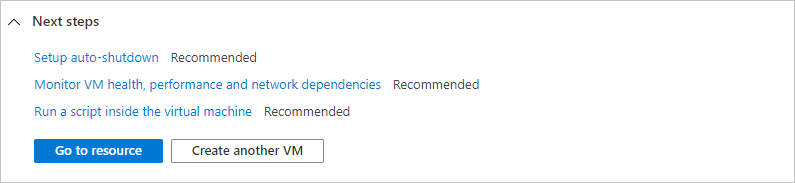
* + - 1. Under **Inbound port rules**, choose **Allow selected ports** and then select **RDP (3389)** and **HTTP (80)** from the drop-down.



* + - 1. Leave the remaining defaults and then select the **Review + create** button at the bottom of the page.

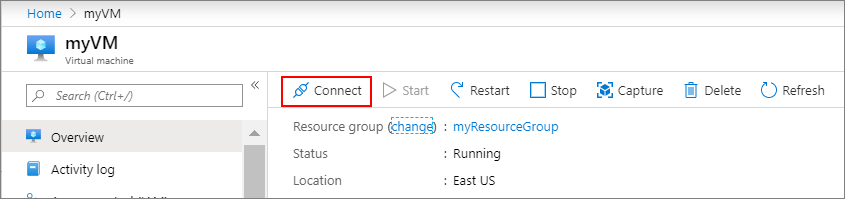


* + - 1. After validation runs, select the **Create** button at the bottom of the page.
      2. After deployment is complete, select **Go to resource**.



Step 2:- Connect to a virtual machine

1. On the overview page for your virtual machine, select the **Connect** > **RDP**.



1. In the **Connect with RDP** page, keep the default options to connect by IP address, over port 3389, and click **Download RDP file**.
2. Open the downloaded RDP file and click **Connect** when prompted.
3. In the **Windows Security** window, select **More choices** and then **Use a different account**. Type the username as **localhost**\*username*, enter the password you created for the virtual machine, and then click **OK**.
4. You may receive a certificate warning during the sign-in process. Click **Yes** or **Continue** to create the connection.

**Conclusion**: Hence we have learned the procedure to launch a virtual machine using trystack.

**Assignment No. 6**

**Aim:** Design and deploy a web application in a PaaS environment.

###### Objective:

* Create simple nodejs app
* Push code to GitHub
* Deploy to Heroku

**Theory:**

# STEP 1: Create simple nodejs app

Create a folder on your local machine and give it a name (of your choice), say MyCoolApp.

Add a file with the name package.json and paste the below content. This file is basic information of our package. (This can also be created by typing command npm init and accepting all default settings.)

{

"name": "coolnodeapp", "version": "1.0.0", "description": "node app ", "main": "app.js",

"scripts": {

"start": "node app.js"

},

"repository": { "type": "git",

"url": ""

},

"author": "",

"license": "ISC", "bugs": {

"url": ""

},

"homepage": ""

}

**package.json**

Add a file, app.js, and paste the below code. This will be the starting point of our app.

Const http = require('http');

const port = process.env.PORT || 3000

const server = http.createServer((req, res) => { res.statusCode = 200;

res.setHeader('Content-Type', 'text/html'); res.end('<h1>Hello World</h1>');

});

server.listen(port,() => {

console.log(`Server running at port `+port);

});

**app.js**

This code is basically opening a port on the local server and serving some HTML.

Save the file and run the below command in the command prompt window (which is open inside the folder):

node app.js

With this, Node will start the server and show the below message:



Now, if we open http://localhost:3000/ in the browser, we will see this:

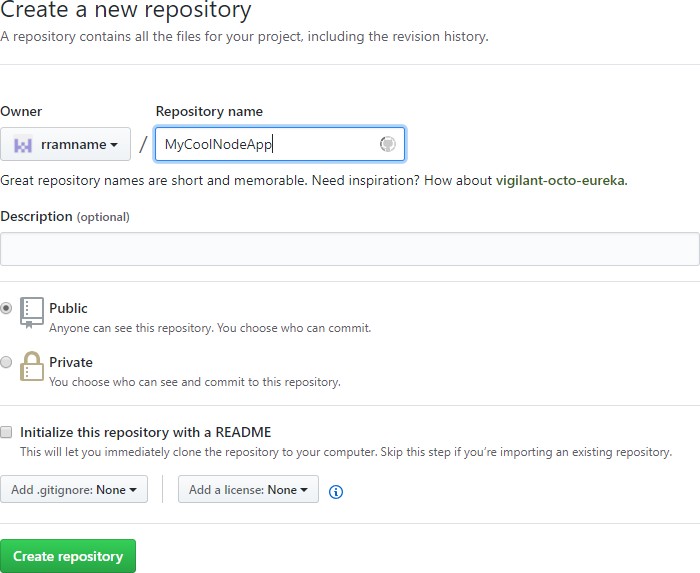


# STEP 2: Push to GitHub

Now want to upload our code to GitHub. This way, we will be able to edit our code from anywhere and also deploy the committed changes to the cloud instantly.

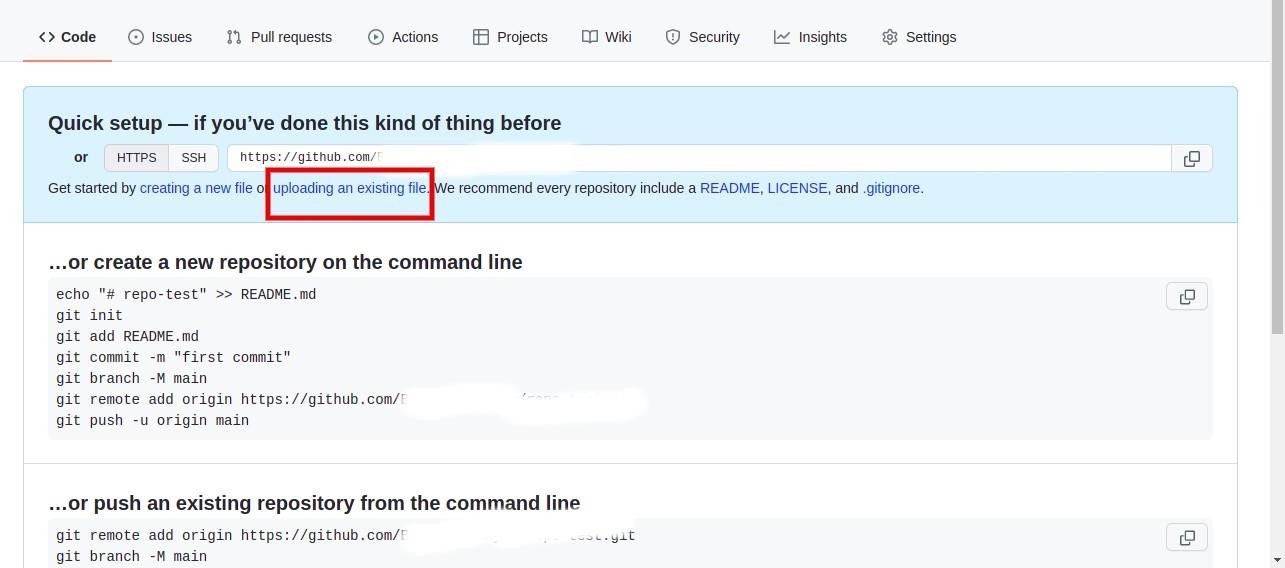
Let’s create a Repository on [GitHub](https://github.com/) by clicking New Repository.

Give it a name, some description, and click Create repository:

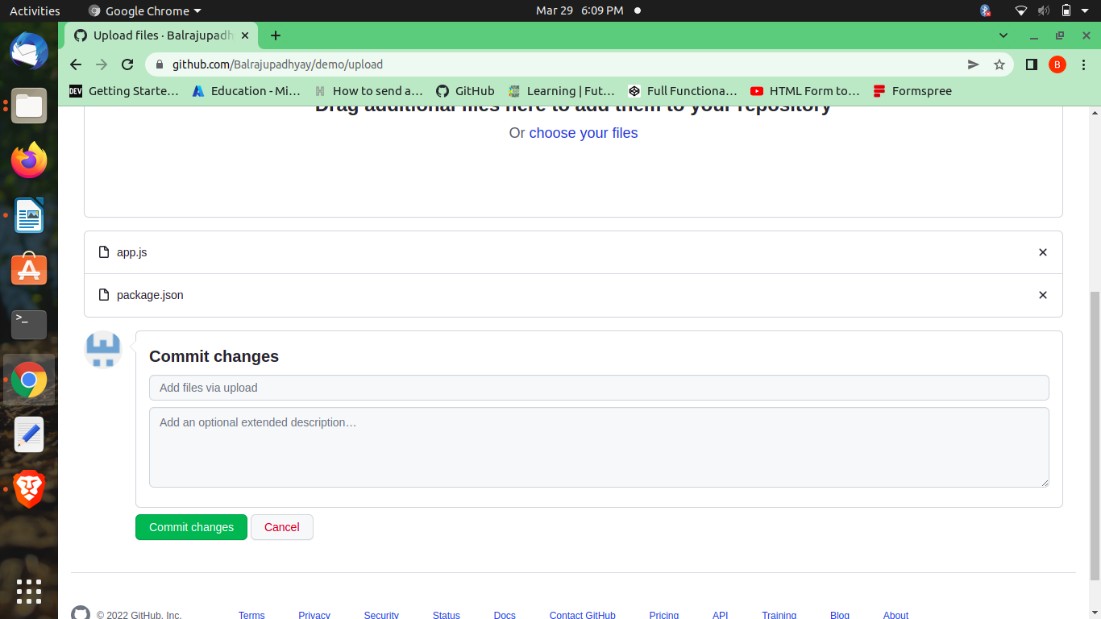


GitHub will create a repository and give you some commands that you can run locally so that you can clone your local folder with your GitHub repository.

Now choose “**upload an existing file”** as shown in figure below :



Drag and drop your previous created file i.e package.json & app.js

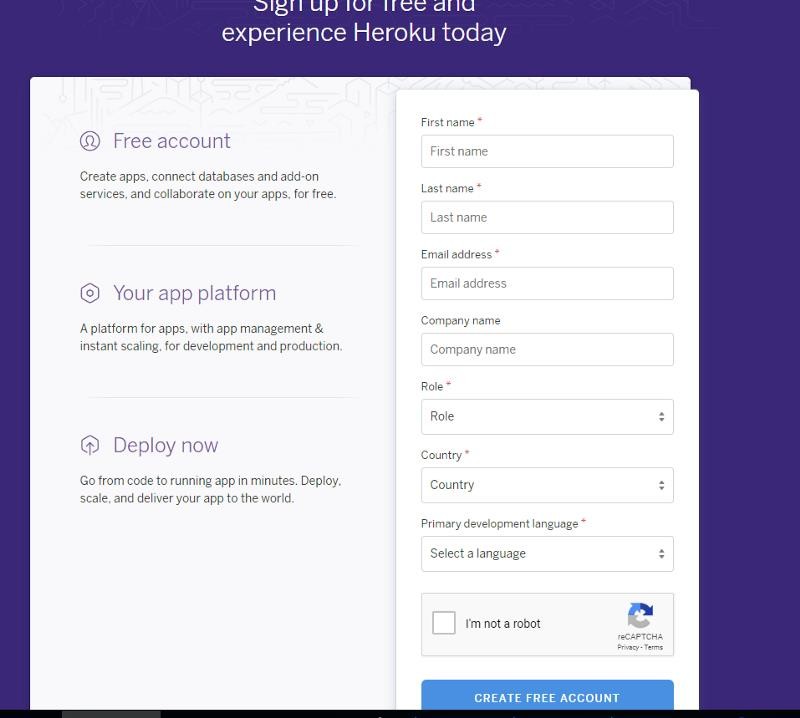


And click on **commit changes .**

# STEP 3: Deploy to Heroku

If you don’t have an account with Heroku, you can open a free one by filling

out this [simple form](https://signup.heroku.com/login).



Once you have your account ready, login with your credentials. Click New on the top right corner and select “Create new app”.

Give your app a name (This will be included in the public URL for your application) and click Create app.

This step will take you to the dashboard of your app. Open Deploy tab and

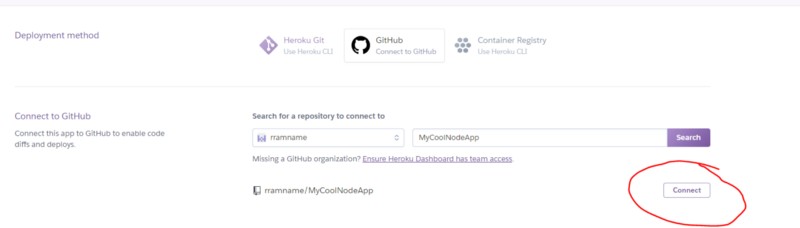
scroll to the “Deployment method” section. Select GitHub as the method.

It will show a “Connect to GitHub” option where we can provide our GitHub

repository. If you are doing it for the first time, Heroku will ask permission to

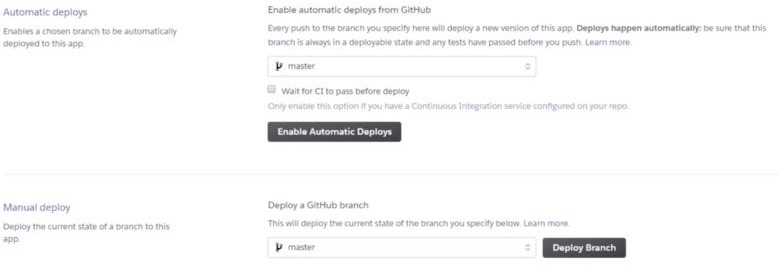
access your GitHub account.

Here, you can search for your GitHub repository and click connect:



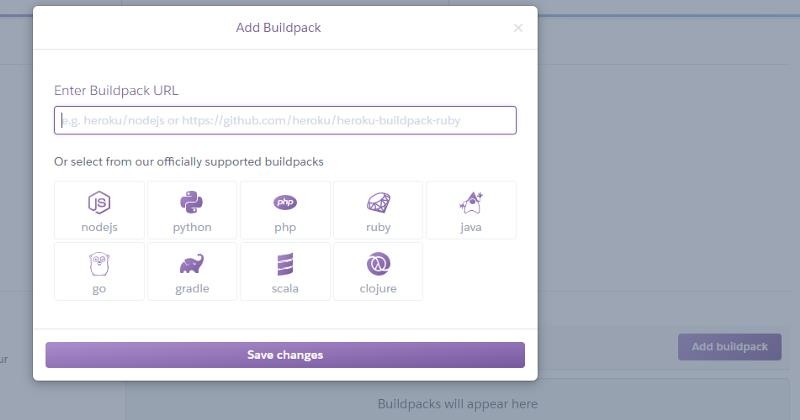
Click “ **Enable Automatic Deploys** “. You can also select the GitHub branch if

you need to, but for this demo we will deploy from the master branch.



Now we need to tell Heroku that our app is a NodeJs app. For that, we will need the NodeJs build back.

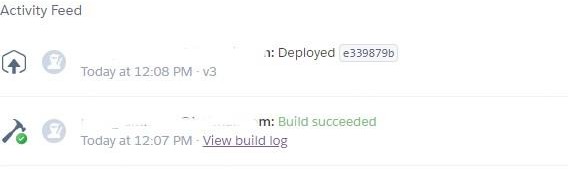
Open the Settings tab and locate Buildpacks and click**“Add buildpack”.**



Select **nodejs** from the options and click Save changes.

Now, go back to the Deploy tab, and click **Deploy Branch** at the bottom.

Heroku will take the code and host it. Open the Activity tab and there you can see the progress:



Open the **settings** tab and scroll down to the **Domains and certificates** section. Here, you can see the URL of your app that was just deployed. Copy and paste that URL in the browser.

**Output:** We just created our own web application that can be accessed over the internet .

**Conclusion:** Learned how to host our own web application in a PaaS environment.

###### FAQ: